

Parallel Computing for Science & Engineering

3/28/2023

OpenMP Review

Instructors:
Lars Koesterke, TACC

General

What are the available Work-Sharing constructs?

Work-Sharing constructs imply a Barrier

Scheduling types

Scoping of variables

- private, shared
- stack variables in functions and subroutines are automatically made private
- be aware of global variables

Other OpenMP constructs

Orphaned worksharing, orphaned OpenMP directives

Important task concepts

Specifics

- 1.) What is the criteria for loop parallelization.
(independent iterations)
- 2.) Are there ways to eliminate dependencies among iterations? (Many times, yes. Clever programming wins in this area.)
3. What are the two main types of worksharing constructs?
(single and for/do)
- 4.) What are **the** two main characteristics of a worksharing construct? (Implied barrier and partitioning of work into tasks.)

Specifics

- 5.) What are the three main types of scheduling for parallel do/for constructs. Be ready to explain the scheduling, costs, what chunk means (all three), and in what cases they would be used. Static (round robin, cheap, data parallel); dynamic (come and get it, not too expensive, imbalanced inner do loops); and guided (same as dynamic, start out big, less expensive than dynamic). Chunk size is the number of iterations assigned to a task, except for guided. For guided the chunk size is the lowest limit of the chunking.
- 6.) Be able to explain the different clauses: shared, private, reduction, firstprivate, lastprivate.
- 7.) Know where implied barriers exist.
- 8.) Be able to determine if a loop can/cannot be parallelized.

Specifics

- 9.) Be able to determine if a code will freeze (e.g. by not all threads executing a worksharing construct).
- 10.) Be able to detect race conditions, and incomplete updates. (Unprivatized temp variable, and not using barrier after a critical region—when needed.)
- 11.) Understand number of threads (executions of) within “replicated regions”, and worksharing regions. -- will be asked to determine number of times a print statement is executed.
- 12.) Be able to spot “common” errors in a simple program. Missing header file, need for private variables, reduction...
- 13.) Understand the operation of an orphaned directive in a subroutine/function.

Specifics

14.) Be able to explain how task (task parallelism) works.

What does the thread-generating task do, after the first task is created? What is the default scoping for global variables and local variables?

15.) Know how to use `omp_get_max_threads`, `omp_get_num_procs`, `omp_get_num_threads`, `omp_get_thread_num`. What is returned inside and outside of a parallel region?

Understanding Replicated & Worksharing Loops

How would this would work (e.g. :N=M=4 = # of threads)

```
!$omp parallel private(j)
do j = 1, N
  !$omp do
  do i = 1, M
    a(i,j) = a(i,j) + j
  end do
end do
```

```
#pragma omp parallel private(j)
for(j=0;j<N;j++){
  #pragma omp for
  for(i=0;i<M;i++){
    a[i][j]= a[i][j] + j
  }
}
```

Don't forget

There was also in introduction

- Why do we do parallel computing
- Scaling: weak and strong
- Amdahl's and Gustavson's law
- Etc.